Report 2: Routy

Alan Goran

September 25, 2017

1 Introduction

The main topic of this assignment was to implement a link-state routing protocol using Dijkstra's shortest path algorithm. This protocol is the same as the one used in routing protocol in Internet routers and it helps us to route messages between logical names. In this assignment, I went through the structure of a link-state routing protocol, maintained a consistent view and reflected on problems related to network failure.

This assignment relates to distributed systems in the sense of communication through routers and nodes, which can pass along messages through hopping from one node to another until it reaches its desired destination.

2 Main problems and solutions

First thing that was implemented in this assignment was a directional map, where I could easily update the map and find nodes that were directly connected to another node. This part was fairly simple and straight forward.

The next function to implement was the Dijkstra algorithm that computes a routing table. A routing table is used to tell us which gateway we should use in order to reach the desired node, for example it tells us that if we want to reach London the shortest path is if we send the message to Madrid. This part was without a doubt the hardest and the main problem of the assignment. While it sounds simple to construct a table that contains all the pathways, it is not simple when we want to use concurrency to update that routing table and make sure that the routing table is always giving us the shortest path for the communication between two places, even when we are adding new nodes with new directional links to the table. This was solved with an iterating function which is looping through all the reachable elements from a given node and adds another node to that path and then updates the sorted list by replacing a given path with a shorter pathway.

After the algorithm was done, an interface was constructed which was described by the symbolic name (London), a process reference and a process identifier. Next, a history function is written to keep track of the link-state messages we send around. The main purpose of this is to avoid cyclic paths and resend messages forever.

Finally the router was implemented to be able to route messages through a network of connected nodes as well as maintaining the view of the network and construct optimal routing tables. The routing process had multiple states, like a symbolic name, a counter, a history of received messages, a set of interfaces, a routing table and a map of the network. To route a message to a destination, the router will check the routing table and find the most optimal gateway, then it will use that gateway's process identifier from the list of interfaces. The main issue here is on the other hand to maintain a consistent view of the networks while the interfaces are added and removed. This issue is solved by having a node collecting a link-state message from all other routers in the network and use that to build a map and use Dijkstra's algorithm to generate a new routing table.

3 Evaluation

I tested my protocol by starting multiple routing processes and connecting them to each other. The test steps are shown below.

First, I started an Erlang node with the command line:

```
erl -name sweden@130.123.112.23 -setcookie routy -connect_all false
```

Then the network was set up using the routy function,

```
routy:start(r1, stockholm),
routy:start(r2, london),
routy:start(r3, mumbai),
routy:start(r4, sydney),
routy:start(r5, vegas).
```

And they were connected to each other according to Fig. 1.

```
r1 ! {add, london, {r2, PIdentifier}},
r1 ! {add, mumbai, {r3, PIdentifier}},
r3 ! {add, sydney , {r4, PIdentifier}},
r3 ! {add, vegas, {r5, PIdentifier}},
r5 ! {add, stockholm, {r1, PIdentifier}}.
```

And after that, the nodes were broadcasted so that the nodes know of each other. The nodes were also updated so that the routing table is updated.

We were now ready to send messages by routing the messages while using the Dijkstra algorithm to provide us with the shortest path. See below for the testing results.

```
> r1 ! {send, sydney, "Hi! What is the weather like? - Stockholm"}.

{send,sydney, "Hi! What is the weather like? - Stockholm"}
stockholm: routing message (Hi! What is the weather like? - Stockholm)
stockholm through: mumbai
mumbai: routing message (Hi! What is the weather like? - Stockholm)
mumbai through: sydney
sydney: received message Hi! What is the weather like? - Stockholm

> r4 ! {send, stockholm, "30 degrees, my man! - Sydney"}.

{send,stockholm,"30 degrees, my man! - Sydney"}
sydney: routing message (30 degrees, my man! - Sydney)
sydney through: stockholm
stockholm: received message 30 degrees, my man! - Sydney
```

We can see from the result above that the algorithm picks the routing path through mumbai which is the shortest path from stockholm to sydney. Another test that I did was to remove the link between stockholm and mumbai so that the message has to go through london and mumbai before reaching sydney.

```
> r1 ! {send, sydney, "Hi ! What is the weather like? - Stockholm"}.

{send,sydney, "Hi ! What is the weather like? - Stockholm"}
stockholm: routing message (Hi! What is the weather like? - Stockholm)
stockholm through: london
london: routing message (Hi! What is the weather like? - Stockholm)
london through: mumbai
mumbai: routing message (Hi! What is the weather like? - Stockholm)
mumbai through: sydney
sydney: received message Hi! What is the weather like? - Stockholm

> r4 ! {send, stockholm, "30 degrees, my man! - Sydney"}.
```

```
{send,stockholm,"30 degrees, my man! - Sydney"}
sydney: routing message (30 degrees, my man! - Sydney)
sydney through: stockholm
stockholm: received message 30 degrees, my man! - Sydney
```

This result is again a proof that the algorithm works since the messages does indeed go through london and mumbai when the link between stockholm and mumbai is removed.

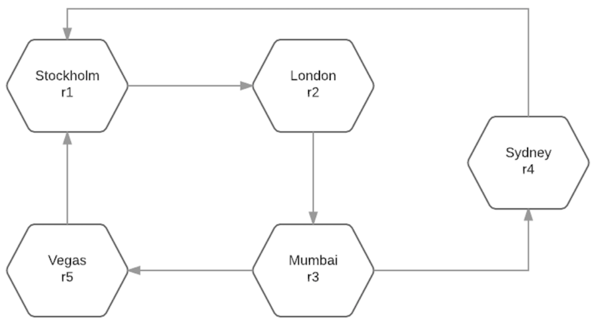

Figure 1: The routing network that I ran the tests on.

4 Conclusions

This assignment was quite hard but it gave a very thorough understanding of OSPF routing protocol. It truly provides you with a new perspective about routers. The results of the assignment proved that the Dijkstra algorithm works perfectly and that routing is done while we successfully held the consistent view of the network.